







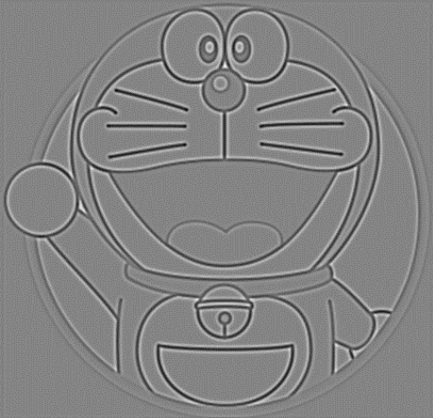

Computer Vision HW1 Report

Student ID: B08505024




Name: 劉虹伶

Part 1.

- Visualize the DoG images for 1.png.

	DoG Image (threshold = 5)		DoG Image (threshold = 5)
DoG1-1.png		DoG2-1.png	
DoG1-2.png		DoG2-2.png	
DoG1-3.png		DoG2-3.png	
DoG1-4.png		DoG2-4.png	

- Use three thresholds (2, 5, 7) on 2.png and describe the difference.

Threshold	Image with detected keypoints on 2.png	
2		
5		
7		

(describe the difference)

Threshold 設定越大，能過標準的 keypoint 數量變少（圖中的紅點數越來越稀疏）。

Threshold=2 時，上方的日文字、盤子與背景的邊緣、湯匙與布丁的接觸面、大布丁與盤子的接觸面都有偵測到 keypoint；Threshold=5 的圖片裡，日文字、盤子上已經只剩下零星紅點，湯匙與布丁的接觸面已經只留下一個點，人臉上的紅點也少了大半。

Threshold=7 更甚，顏色對比需要很大（黑與白）才能被偵測為 keypoint，紅點只出現在眼睛、嘴巴、大小布丁的焦糖邊緣上，盤子邊緣、日文字上已經沒有任何紅點。

	Threshold=2	Threshold=5	Threshold=7
keypoints	199	52	23






Part 2.

- Report the cost for each filtered image.

Gray Scale Setting	Cost (1.png)
cv2.COLOR_BGR2GRAY	1207799
$R*0.0+G*0.0+B*1.0$	1439568(Highest)
$R*0.0+G*1.0+B*0.0$	1305961
$R*0.1+G*0.0+B*0.9$	1393620
$R*0.1+G*0.4+B*0.5$	1279697
$R*0.8+G*0.2+B*0.0$	1127913(Lowest)

Gray Scale Setting	Cost (2.png)
cv2.COLOR_BGR2GRAY	183850
$R*0.1+G*0.0+B*0.9$	77884(Lowest)
$R*0.2+G*0.0+B*0.8$	86023
$R*0.2+G*0.8+B*0.0$	188019(Highest)
$R*0.4+G*0.0+B*0.6$	128341
$R*1.0+G*0.0+B*0.0$	110862

- Show original RGB image / two filtered RGB images and two grayscale images with highest and lowest cost.


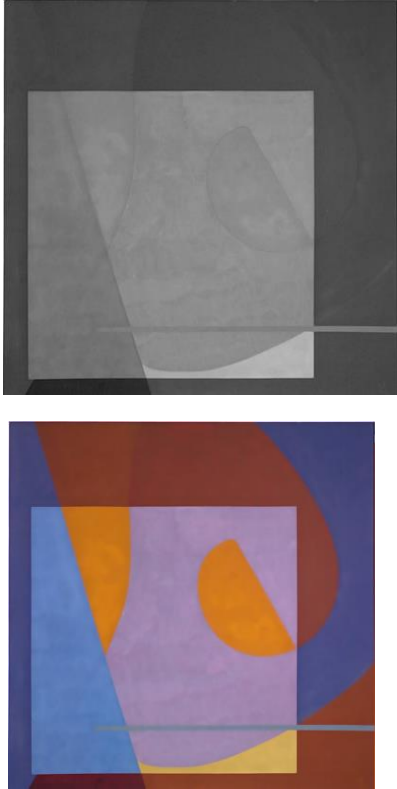
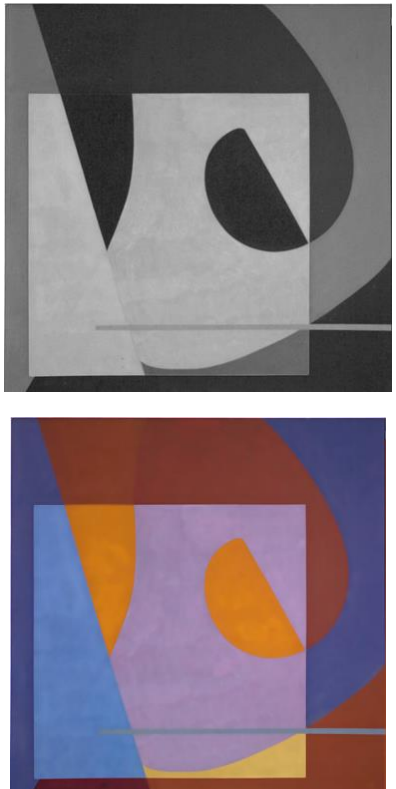
Original RGB image (1.png)	Filtered <u>RGB image</u> and <u>Grayscale image</u> of Highest cost	Filtered <u>RGB image</u> and <u>Grayscale image</u> of Lowest cost
	 	 

(Describe the difference between those two grayscale images)

$R*0.0+G*0.0+B*1.0$ 是 cost 最大的 (1439568)，而 $R*0.8+G*0.2+B*0.0$ 最小 (1127913)，cost 越小代表對應的 Gray Scale 是更好的 guidance image。

觀察原圖會發現，只有草地縫隙間的一些黑色、雜草的土黃色、楓葉的斑點以外，就是由大區塊的綠色 (G) 與楓葉的紅色 (R) 所組成。guidance image 要好的話就需與原圖的相似度高，cost 最高的 $R*0.0+G*0.0+B*1.0$ 以 B 為唯一的參數，不採用紅 (R)、綠 (G) 的 value，會省略太多重要資訊， $R*0.0+G*0.0+B*1.0$ 的 Grayscale image 中也可看出來圖片變的漆黑，較難看出圖像輪廓。

再觀察 $R*0.8+G*0.2+B*0.0$ 的係數，改成不採用藍色 (B)，以紅綠去 Grayscale 的效果很好，保留了圖像的輪廓，也能明顯看出楓葉與草地的明暗有所不同。

Original RGB image (2.png)	Filtered <u>RGB image</u> and <u>Grayscale image</u> of Highest cost	Filtered <u>RGB image</u> and <u>Grayscale image</u> of Lowest cost
		

(Describe the difference between those two grayscale images)

$R*0.2+G*0.8+B*0.0$ 是 cost 最大的 (188019)，而 $R*0.1+G*0.0+B*0.9$ 最小 (77884)。

同樣先觀察原圖的色塊 (R,G,B)：大面積的有藍色 (66,67,115)、紅色 (124,56,56)、淡紫色 (157,131,170)；以及小面積的兩塊黃色 (206,121,25)，若保留圖片資訊，把 B、R 的係數調大一些會比較好。

Cost 最高的 $R*0.2+G*0.8+B*0.0$ ，G 的係數是 0.8，R、B 的係數是 0.2 和 0.0，故出來的 Grayscale image 幾乎讓外框的顏色一致、框內的區塊也變得平淡。反之觀察 cost 最小的 $R*0.1+G*0.0+B*0.9$ ，成功保留圖片的圖樣，仔細看也會發現 cost 最小的 Filtered RGB image 正方形方框以外的邊緣更清晰一些。

(RGB 查找網址：https://www.ginifab.com.tw/tools/colors/color_picker_from_image.php)

- Describe how to speed up the implementation of bilateral filter.

我沒有特別改成用別的方式 implementation(ex. Gaussian by FFT approach or iterative box filter)，只有盡量減少使用 for loop，除了 construct Spatial kernel 的 for loop 以外，只用了三層 for loop 去跑圖片(h*w*channel)，其餘都用 np 的 function(ex. np.divide、np.square、np.multiply、.sum(axis=1))去跑。

若要更快的話。改成 Gaussian by FFT approach 之類的 $O(1)$ 方法先處理 Gaussian blur 可以減少 np.multiply 的次數。